

# 1 Interpolation Code

A code written in Python, that calculates the required cross sections and their respective moments i.e.  $\sigma_{total}$ ,  $\sigma_{-2}$  &  $\sigma_{-1}$  is presented below. Values assigned to parameters in the code below are those of  $^{138}\text{La}$  and can be easily changed according to the desired output.

The energies and cross sections data are loaded in arrays whose lengths are dependent on the region of interest and experimental limitations. A one dimensional cubic interpolation function was defined, one dimensional implying there is one independent variable,  $E_\gamma$  in this case. Data reproduced by the interpolating function is stored in a file named *output\_data\_138La.dat*. One can execute this code in terminal or Python GUI by navigating to the directory with the .py file and use the command: *python filename.py*. The integration limits can be controlled in the *xint* array. The main purpose of this program is to calculate total photonuclear cross sections and their moments, using cubic-spline interpolation, where the interpolant is a piecewise function of cubic polynomials generated between a pair of consecutive data points. This was done using a numerical integration method. Minimizing the increments of energy,  $h$  in Eq. 1, to order  $10^{-3}$  MeV reduces the error (systematic) of integration to negligible values, the error is determined from the cross section values yielded by the interpolation function. The output can be represented in graphs see Fig. 1.

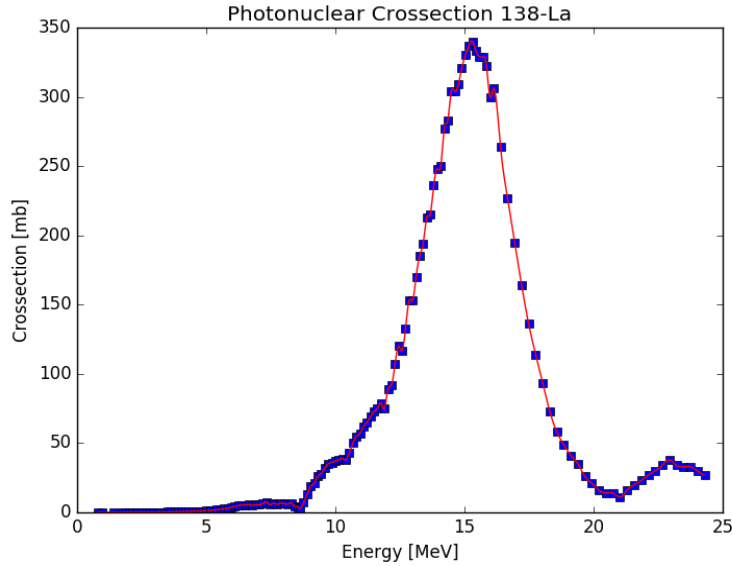


Figure 1: Photonuclear cross section of  $^{138}\text{La}$  showing the interpolation function  $f(E_\gamma)$ , red solid line.

The error  $d\sigma$  of the CTR can be obtained as a difference between the numerical result of the integral and the CTR integral value.

$$d\sigma = \int_{E_{\gamma(min)}}^{E_{\gamma(max)}} \sigma(E_{\gamma}) dE_{\gamma} - h \left[ \frac{\sigma(E_{\gamma(min)}) + \sigma(E_{\gamma(max)})}{2} + \sum_{k=1}^{N-1} \sigma(E_{\gamma(min)} + kh) \right] \quad (1)$$

where the cross section  $\sigma(E_{\gamma})$  values are generated by the interpolant,  $h$  is the energy increment and  $N$  is the number of trapezoids.

```

### Libraries

from numpy import arange
from scipy.interpolate import interp1d
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import trapz
import numpy
import math

### DATA INPUT

x = np.array([])                    #Energy [MeV]
y = np.array([])                    #Cross-section [mb]

### INTERPOLATION & INTEGRATION

f = interp1d(x,y,kind = 'cubic')    # Cubic spline, Interpolating function

xint = np.arange(0.800,27.84,0.001) # Energy array
yint = f(xint)                       # Cross sections by the Interpolating func.
yint2 = f(xint)/(xint**2)            # E1_(-2)sigma data
f_X1 = (1/((1)*(math.pi**2)*(197.326)**2))*(yint/10*xint) ## photon strength function

with open("output_data_138La.dat", "w") as out_file: # Saving data into a file: Cross section
    for i in range(len(xint)):
        output_string = ""
        output_string += str(xint[i])
        output_string += " " + str(yint[i])    # storing data in column format
        output_string += "\n"                 # new line
        out_file.write(output_string)
        #print output_string                  # Printing Interpolation data(optional)

with open("gsf_output_data_138La.dat", "w") as out_file: ## Gamma-Strength functions file
    for i in range(len(xint)):
        output_string = ""
        output_string += str(xint[i])
        output_string += " " + str(f_X1[i])
    output_string += "\n"
    out_file.write(output_string)
    #print output_string                      ## Printing Interpolation data

## NUMERICAL INTEGRATION

h = 0.001
summ = 0

for i in range(0,len(xint)):
    summ = summ + yint2[i]

summ2 = (yint2[0] + yint2[len(xint)-1])*0.5

```

